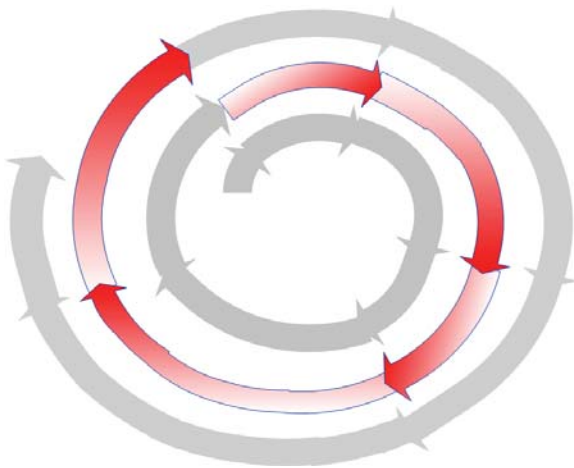


## ITERATION 2: OVERVIEW

### FUNCTIONAL DELIVERY



**Bite:** Identify functional requirements to meet business objectives and then develop, implement and deploy a system that meets most of the functional requirements.

**Time:** 40-50% of development time

**Input:** Proposal, Management Document

**Process evidence:** Functional system (annotated screen shots), evidence of decision making

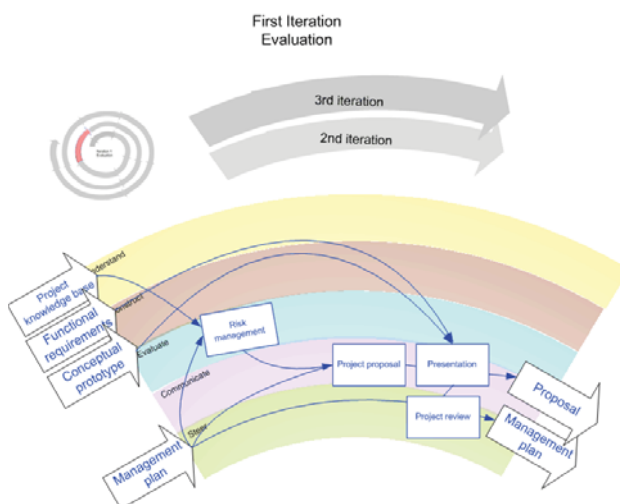
**Client:** Discussion, testing, deployed system, formal letters

The second iteration is the heart of the development process. In it we progress from having a high level understanding of the business problem/opportunity to having a system deployed in active use by users.

In the first iteration we have:

- established a development team
- established firm communication links with the client
- established that we are on common ground with the client about the high level need for the project
- established a methodology and work practices.

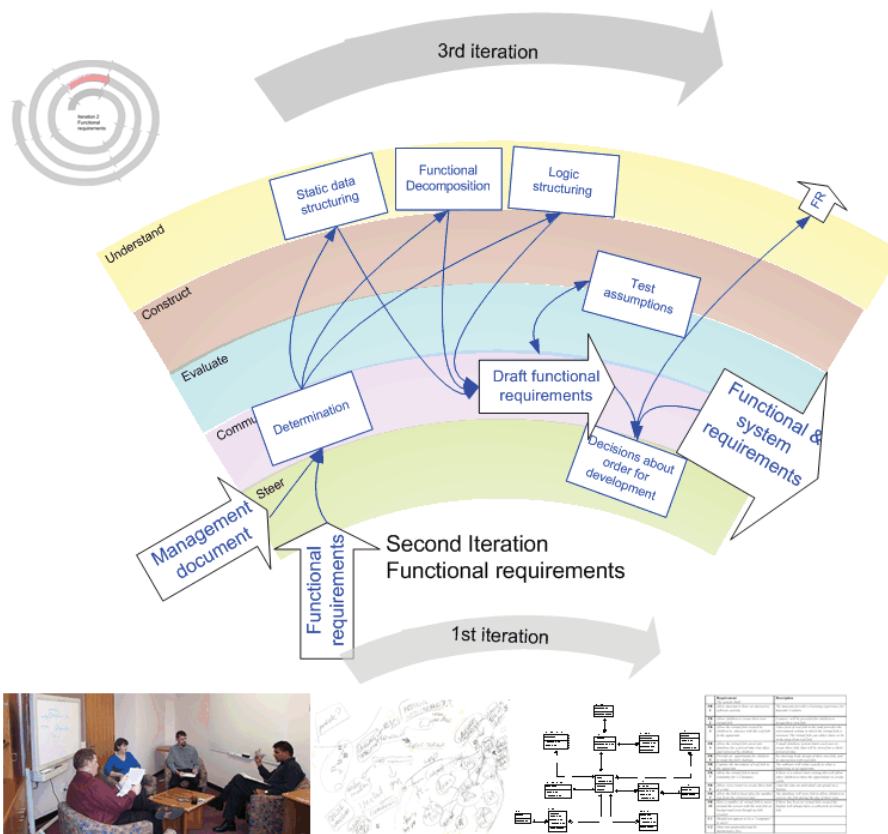
In the second iteration we progress through a development cycle. We first work out in some detail what needs to be done, work out how to do it, and (you guessed it) do it. We finish this iteration with a deployed system that functions and is useful. Exactly what that system does (and what gets left until the third iteration) we'll come back to. For the moment, it is important that we know that the date of delivery is the most important variable. This is called a timebox, which means that we adjust what we are trying to do to fit the time and resources available.



Remember that the internals of each of these sectors is not explicitly defined. What is more important is the flow of information as we massage and enhance our understandings as we move to the output of that sector. We should, though, always be able to explain why we are doing something, and demonstrate how it helps us towards the goal.

Although the focus here is on structured processes, we undertake them according to agile principles. We expect that scrum meetings are the basis for each day's work, that there is a focus on quality and simplest thing that could possibly work, and that the emphasis remains on outcomes rather than processes for the sake of it. Always keep a metaphorical eye on the rainbow for that sector to a) check

## Second Iteration Functional requirements



You need to understand

- Business objectives that drive how the work is done
- Information people need to do their jobs
- Data (definition, volume, size etc) handled within the organisation to support the jobs
- When, how and by whom, are the data moved, transformed and stored
- Sequence and other dependencies among different data handling activities
- Rules governing how the data are handled and processed
- Policies and guidelines that describe the nature of the business and the market and environment in which it operates
- Key events affecting data values and when these events occur.

that all workstreams are covered, and b) know how what you are doing directly feeds the outcome of that sector. We should include the client as much as possible through the process and will be sending formal documentation at the end of each sector (sometimes a detailed proposal, sometimes an update letter).

The evaluation sector is the transition between iterations. In it, we look back at what we have done so far and plan the next sectors. This is an important phase in formalising our interaction with the client by means of a proposal.

## Functional Requirements

The Functional Requirements sector can be summarised in one word: “what”. Here we work out what the system needs to do (as opposed to how to do it).

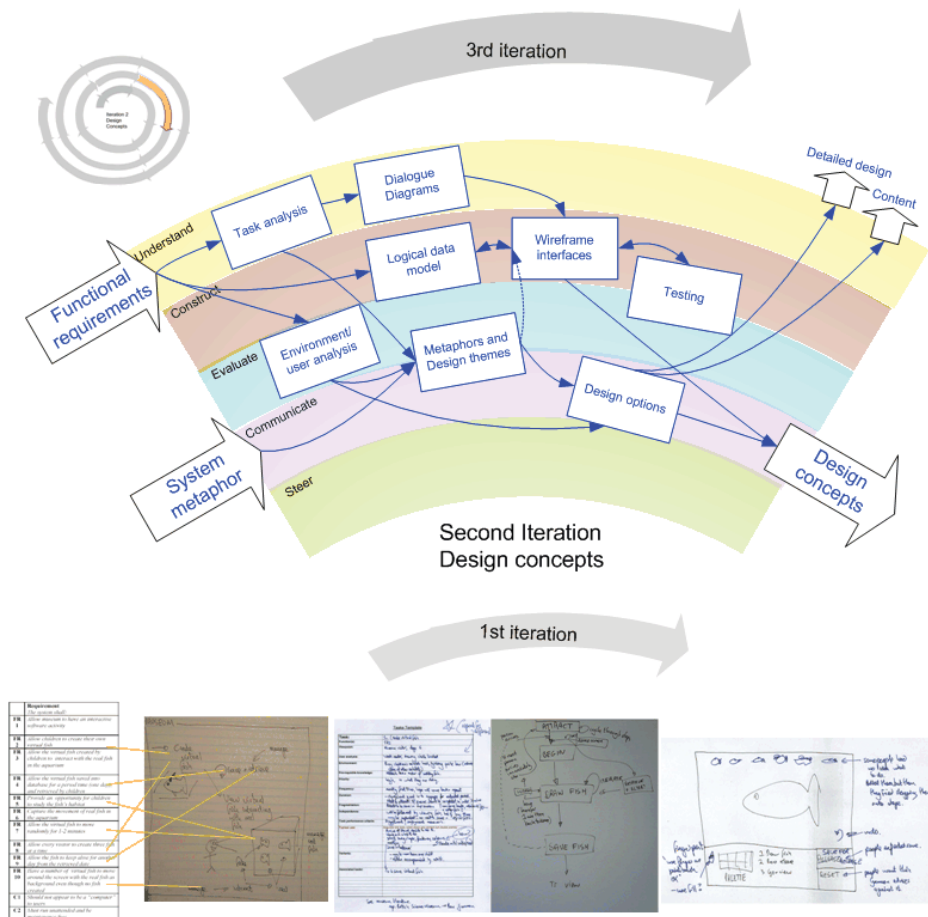
The functional requirements are instructions describing what functions the software is supposed

to provide, what characteristics the software is supposed to have, and what goals the software is supposed to meet or to enable users to meet.

Functional requirement statements begin with “The system shall...”. (Note: there are some moves to replace this with user stories “The user shall...”). They are usually accompanied by system requirements: “The system shall have....”.

To get to these functional requirements we go through a process of collecting and understanding as much information as we can. Very quickly we have to understand every detail of the organisation we’re working with. From the first interview we have to be knowledgeable about the business. Fortunately we have some techniques to help collect and structure this information, working from both above and below the problem. We use functional decomposition to help us understand the business workflow, data modelling to identify structures and relationships and logic structuring to capture business rules. We also observe and question users.

## Second Iteration Design concepts



These techniques will lead us directly to directly to functional requirements that define what the system has to do. We test these functional requirements to make sure that they address the business opportunity and are feasible to develop (though we have to be careful not to restrict our thinking). These requirements become the basis for our development and are fed onto the next sector.

## Design Concepts

In the Design Concepts we develop the look and feel of a system that meets the functional requirements. Although we mostly think about interfaces being computer screens, most projects have other non-screen interface elements, for example forms, reports and sometime s physical interactivity systems.

The output of this sector is a design proposal that we will formally present to the client. The client and users should be closely involved in this sector.

This is the most fun and creative stage of the whole development process, but it is also a sector that requires much rigour and care. All five workstreams are likely to be running simultaneously and you should expect to have several balls in the air at the same time.

Four tricks this time:

- **Test everything:** users don't behave or think the way you do

- **Don't cheat:** building an interface in your favorite IDE and then working backwards through process doesn't work. Do not skimp on the dialogue diagrams.

- **Users should forget they are using your system:** the measure of success is primarily in match to the users' model of how something should be done, and people like things simple so you should be as **elegant as possible:** a few well designed interfaces are far preferable to a multitude of hotchpotch.

At the same time as are doing this we work on developing design themes. Starting with the metaphor, we come up with some alternatives about what the

We present the results of this development and testing in Design Concepts presentation/document to the client.

By the end of this sector we have a blueprint to give to programmers. They should be able to build exactly





what we specify.

The design specification integrates both the look and the how. We take the design theme and wireframe and develop detailed interface specifications and complete designed products such as the user manual. Alongside this we continue to develop the database (the key here is 'no magic' - all data come from and goes to somewhere). The physical data model is a detailed specification of the information to be stored and manipulated by our system.

Alongside the 'how' of the interfaces we also work out the how of the back end. This starts with the database and system architecture (expressed in a diagram), then decisions about coding structures and ends with development and testing of a stable development platform.

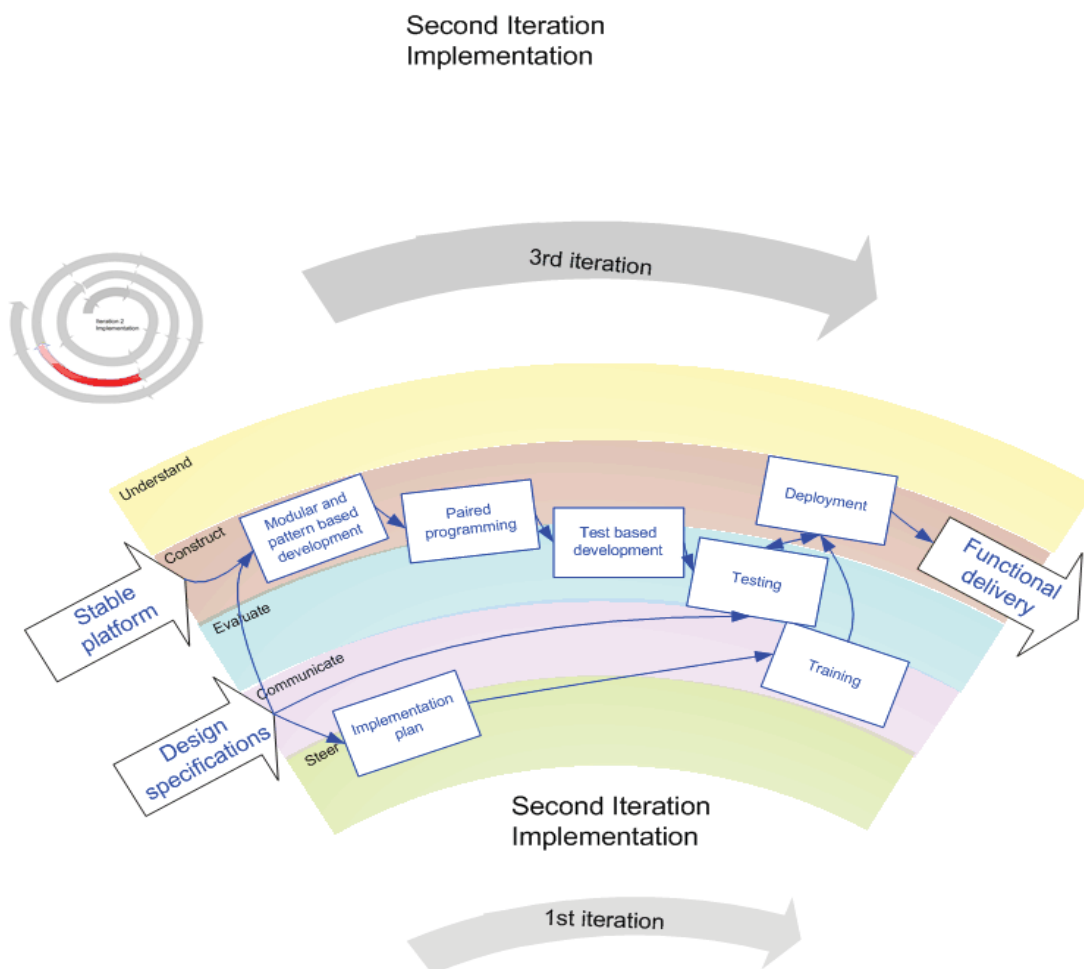
The stable development platform is the framework upon which your system is to be built. For example, for a database with a web front end we would expect you to demonstrate connectivity and basic database functions via the web (insert, delete, query, update) plus any standard infrastructure (login etc).

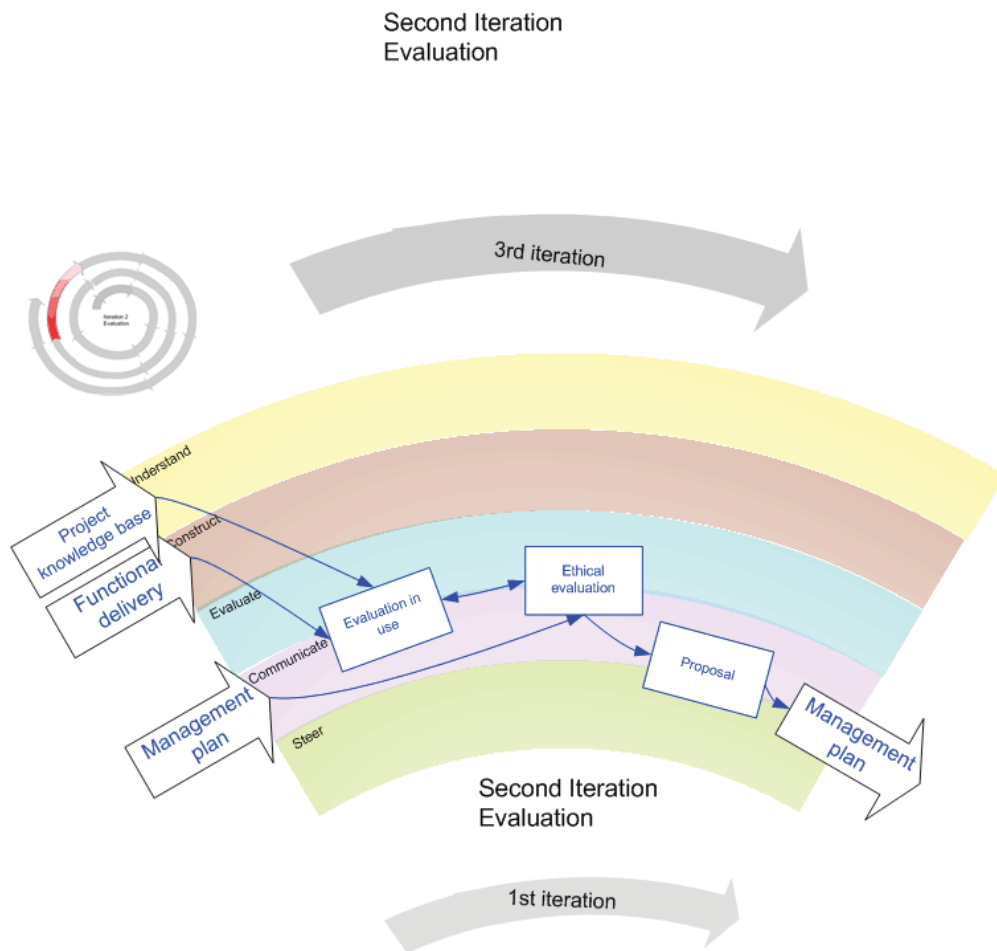
## Implementation

The implementation stage is about building a system according to the specifications. This is where you get the pay back for all the hard work you've done so far. Building upon the stable platform, with a physical database designed and tested, and with detailed interface specifications, this implementation shouldn't be an enormous task. Fortunately, software engineering also facilitates the implementation sector. We use several tools from Extreme Programming (XP) to help us.

We use modular and pattern based development, we use paired programming, and we use test based development. We test extensively and then we test some more. We carefully think out the best way of deploying the system and we do it. While this is happening we train the users.

At the end of this sector we have deployed a system that works and, importantly is in active use by the users.





## Evaluation

Once the system has been in use for a while we need to take time to do an evaluation. This is more than just a “does it work?” (although that is important too, but also includes a hard look at the functional requirements (were they right?, are there any missing?, and so on) and a reflective look at all aspects of the development. The client should be closely involved in this reflective process.

